```
#importing necessary libraries
import numpy as np
import pandas as pd
```

Reading data

```
#view of the data set
data = pd.read_csv('abcnews-date-text.csv')
data.head()
```

	publish_date	headline_text
0	20030219	aba decides against community broadcasting lic...
1	20030219	act fire witnesses must be aware of defamation
2	20030219	a g calls for infrastructure protection summit
3	20030219	air nz staff in aust strike for pay rise
4	20030219	air nz strike to affect australian travellers

Data Pre-processing

```
#size and shape of the data set
data.shape
```

```
(1226258, 2)
```

```
data.describe()
```

	publish_date
count	1.226258e+06
mean	2.010875e+07
std	4.720924e+04
min	2.003022e+07
25%	2.007052e+07
50%	2.011051e+07
75%	2.014113e+07
max	2.020123e+07

```
data.isnull().sum()
```

```
publish_date     0
headline_text    0
dtype: int64
```

```
data.dtypes
```

```
publish_date      int64
headline_text    object
dtype: object
```

Feature Engineering

```
#creating date, year and month as a seperate columns from publish_date
data["year"] = data["publish_date"].astype(str).str[:4].astype(np.int64)
data["month"] = data["publish_date"].astype(str).str[4:6].astype(np.int64)
data["date"] = data["publish_date"].astype(str).str[6:].astype(np.int64)
data.head()
```

	publish_date	headline_text	year	month	date
0	20030219	aba decides against community broadcasting lic...	2003	2	19
1	20030219	act fire witnesses must be aware of defamation	2003	2	19
2	20030219	a g calls for infrastructure protection summit	2003	2	19
3	20030219	air nz staff in aust strike for pay rise	2003	2	19
4	20030219	air nz strike to affect australian travellers	2003	2	19

```
#unique months in the dataset
data.year.unique()
```

```
array([2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013,
       2014, 2015, 2016, 2017, 2018, 2019, 2020])
```

```
data.month.unique()
```

```
array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,  1])
```

```
#creating another column word_count to understand how many words in each headline_text
data["word_count"] = data["headline_text"].str.len()
data.head()
```

	publish_date	headline_text	year	month	date	word_count
0	20030219	aba decides against community broadcasting lic...	2003	2	19	50
1	20030219	act fire witnesses must be aware of defamation	2003	2	19	46
2	20030219	a g calls for infrastructure protection summit	2003	2	19	46

```
# Creating character count, average word length, punctuations count, stopword count columns f
data['char_count'] = data['headline_text'].apply(lambda x: len(str(x)))

data['mean_word_length'] = data['headline_text'].apply(lambda x: np.mean([len(w) for w in str

import string
data['punctuation_count'] = data['headline_text'].apply(lambda x: len([c for c in str(x) if c


#removing the stopwords
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer,PorterStemmer
from nltk.tokenize import word_tokenize

import string
```

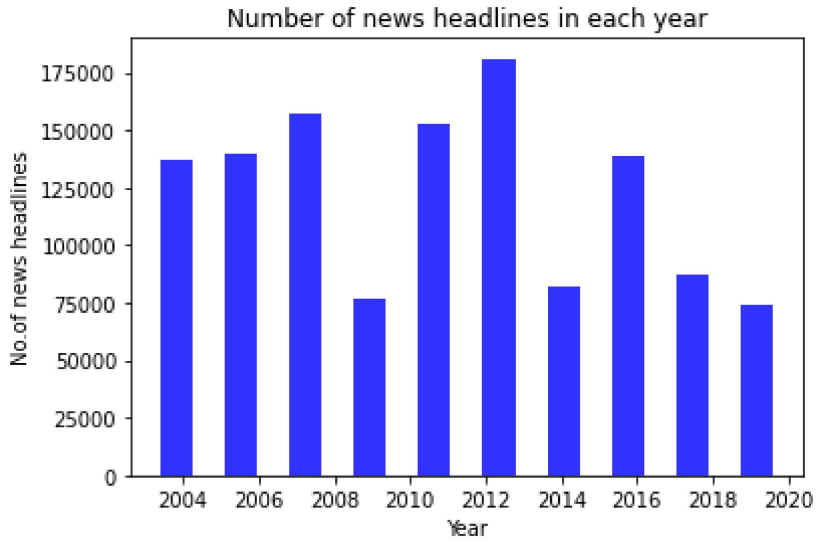
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
#showing the stopword count and all the character counts in each headline ex:- punctuation co
english_stops= set(stopwords.words('english'))
data['stop_word_count'] = data['headline_text'].apply(lambda x: len([w for w in str(x).lower(
data.head()
```

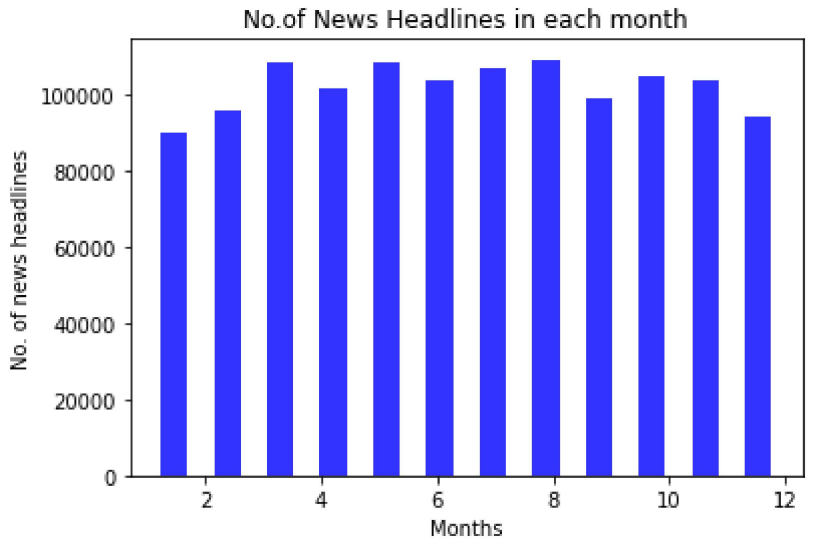
	publish_date	headline_text	year	month	date	word_count	char_count	mean_word_le
0	20030219	aba decides against community broadcasting lic...	2003	2	19	50	50	7.50
1	20030219	act fire witnesses must be aware of defamation	2003	2	19	46	46	4.87
		a g calls for						

Visualization the features extracted

```
# Visualizing how many articles where published every year, month and each day
import matplotlib.pyplot as plt
plt.hist(data['year'], facecolor='blue', alpha=0.8, rwidth = 0.5)
plt.xlabel('Year')
plt.ylabel('No.of news headlines')
plt.title('Number of news headlines in each year')
plt.show()
```

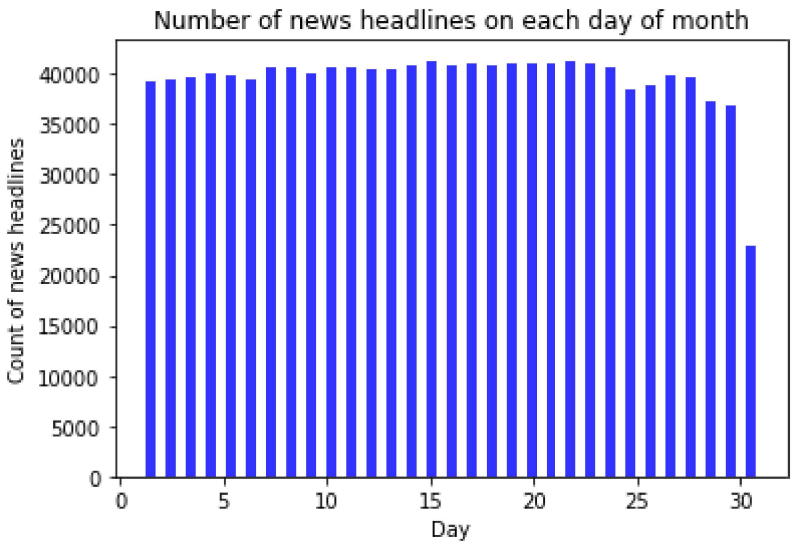



```
plt.hist(data['month'],12, facecolor='blue', alpha=0.8, rwidth = 0.5)
plt.xlabel('Months')
plt.ylabel('No. of news headlines')
plt.title('No.of News Headlines in each month')
plt.show()
```



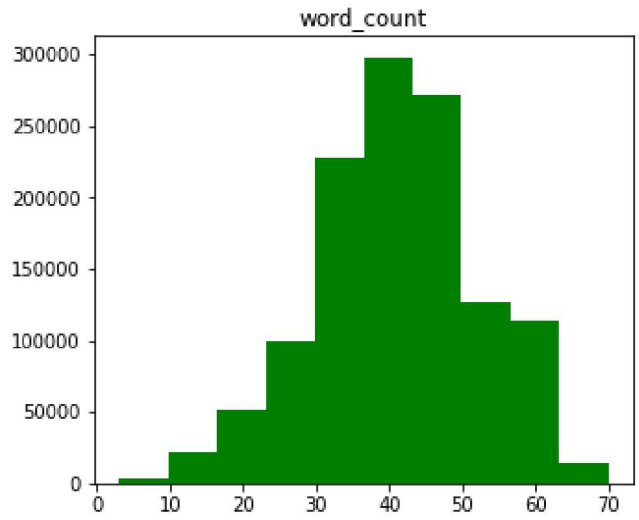

```
plt.hist(data['date'],31, facecolor='blue', alpha=0.8, rwidth = 0.5)
plt.xlabel('Day')
plt.ylabel('Count of news headlines')
```

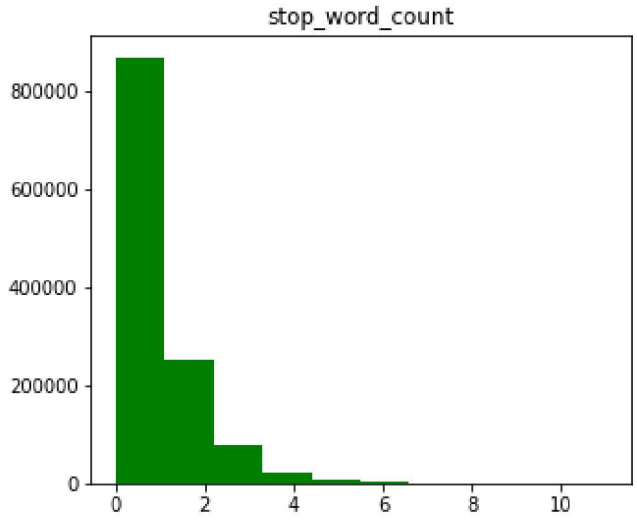
```
plt.title('Number of news headlines on each day of month')
plt.show()
```

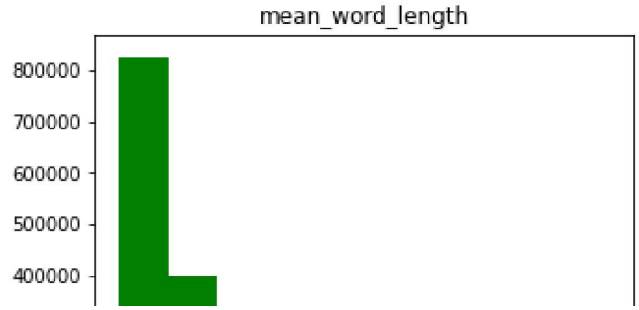


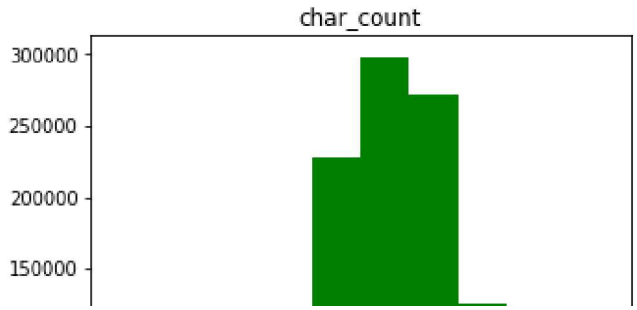

```
# defining the new features in a list
features = ['word_count', 'stop_word_count', 'mean_word_length', 'char_count', 'punctuation_c


# plotting histograms for the new features
data[features].hist(figsize = (12,16), color='green', grid=False, bins=10)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fb81656ce50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fb8164ebb90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fb8164af1d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fb816466790>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fb81641add0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fb8163e08d0>]],
      dtype=object)
```









Sentiment Analysis


```
#Creating a corpus from the headline_text
corpus = str()
for i in range(len(data['headline_text'])):
    corpus += (' ')+data['headline_text'][i]
```

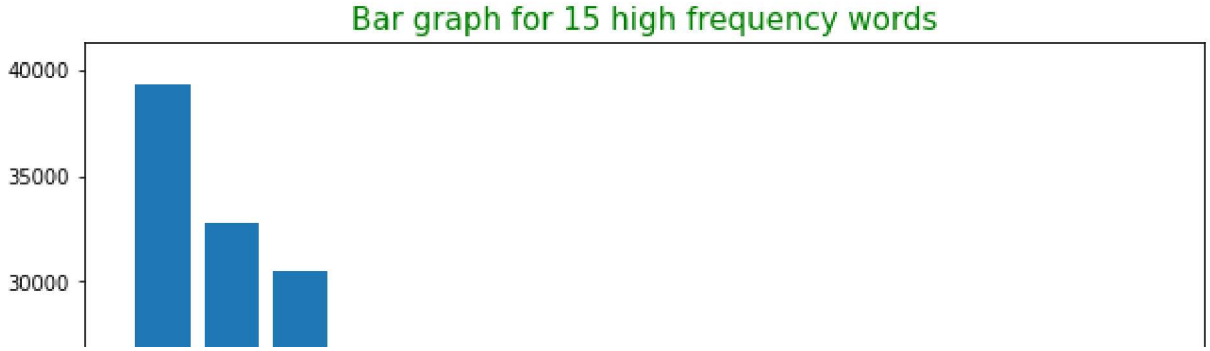
```
# Toenization
import nltk
nltk.download('punkt')
words = nltk.word_tokenize(corpus)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
# removing stopwords and punctuations
f words = [w for w in words if not w in english stops]
```

```
punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
fp_words = [w for w in f_words if not w in punctuations]


# To find the frequency of clean words
freq = nltk.FreqDist(fp_words)

for key,val in freq.items():
  str(key) + ':' + str(val)




#removing words whose frequency is less than 500
for x in list(freq.keys()):
  if freq[x]<=500:
    del freq[x]

# finding top 15 commonly used words
freq_2=freq.most_common(15)
print(freq_2)
words=[x[0] for x in freq_2]
word_freq=[x[1] for x in freq_2]

# plotting a graph for top 15 used words.
plt.figure(figsize=(10,8))
plt.bar(words, word_freq)
plt.title('Bar graph for 15 high frequency words', fontsize=15, color='Green')
plt.xlabel('Words', fontsize=15, color='Green')
plt.ylabel('word_frequnecy', fontsize=15, color='Green')
plt.show()
```

```
[('police', 39384), ('new', 32824), ('man', 30512), ('says', 23014), ('us', 20045), ('cc
```



Emotional Analysis

```
#importing few required libraries
!pip install tweet-preprocessor 2>/dev/null 1>/dev/null
import preprocessor as p
import numpy as np
import pandas as pd
import keras
from sklearn.model_selection import train_test_split
import tensorflow as tf
from keras.models import Sequential
from keras.layers.recurrent import LSTM, GRU,SimpleRNN
from keras.layers.core import Dense, Activation, Dropout
from keras.layers.embeddings import Embedding
from keras.utils import np_utils
from sklearn import preprocessing, decomposition, model_selection, metrics, pipeline
from keras.layers import GlobalMaxPooling1D, Conv1D, MaxPooling1D, Flatten, Bidirectional, Sp
from keras.preprocessing import sequence, text
from keras.callbacks import EarlyStopping
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
import plotly.graph_objects as go
import plotly.express as px
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint
from tqdm.notebook import tqdm
from tqdm import tqdm


#data preparation, taking the data set for cleaning purpose for sentiment analysis
data1 = pd.read_csv("text_emotion.csv")
misspell_data = pd.read_csv("text_emotion.csv",sep=":",names=["correction","misspell"])
misspell_data.misspell = misspell_data.misspell.str.strip()
misspell_data.misspell = misspell_data.misspell.str.split(" ")
misspell_data = misspell_data.explode("misspell").reset_index(drop=True)
misspell_data.drop_duplicates("misspell",inplace=True)
miss_corr = dict(zip(misspell_data.misspell, misspell_data.correction))

#Sample of the dict
```

```
{v:miss_corr[v] for v in [list(miss_corr.keys())[k] for k in range(20)]}
```

```
{'#topicmaps': '1956972116,neutral,jansc,No Topic Maps talks at the Balisage Markup Conf
 '(via': '1956972116,neutral,jansc,No Topic Maps talks at the Balisage Markup Conference
 '//plurk.com/p/wxidk': '1956975876,neutral,jubaldo,feels strong contractions but wants
 '//tr.im/mL6Z': '1956972116,neutral,jansc,No Topic Maps talks at the Balisage Markup Co
 '//www.djhero.com/': '1956968636,worry,mcsleazy,Hmmm. http',
 '@bobdc)': '1956972116,neutral,jansc,No Topic Maps talks at the Balisage Markup Confere
 'BC': '1956968477,worry,xxxPEACHESxxx,Re-pinging @ghostridah14',
 'bf': '1956968477,worry,xxxPEACHESxxx,Re-pinging @ghostridah14',
 "didn't": '1956968477,worry,xxxPEACHESxxx,Re-pinging @ghostridah14',
 'down': '1956968636,worry,mcsleazy,Hmmm. http',
 'friends': '1956968477,worry,xxxPEACHESxxx,Re-pinging @ghostridah14',
 'go': '1956968477,worry,xxxPEACHESxxx,Re-pinging @ghostridah14',
 'is': '1956968636,worry,mcsleazy,Hmmm. http',
 'like': '1956968477,worry,xxxPEACHESxxx,Re-pinging @ghostridah14',
 'my': '1956968477,worry,xxxPEACHESxxx,Re-pinging @ghostridah14',
 nan: 'tweet_id,sentiment,author,content',
 'prom?': '1956968477,worry,xxxPEACHESxxx,Re-pinging @ghostridah14',
 'to': '1956968477,worry,xxxPEACHESxxx,Re-pinging @ghostridah14',
 'why': '1956968477,worry,xxxPEACHESxxx,Re-pinging @ghostridah14',
 'you': '1956968477,worry,xxxPEACHESxxx,Re-pinging @ghostridah14'}
```

```
#using this function we are removing the misspllled words ex:- 'acord': 'accord' (few are list
def misspelled_correction(val):
    for x in val.split():
        if x in miss_corr.keys():
            val = val.replace(x, miss_corr[x])
    return val
```

```
#cleaning the data i.e making corrections to the mispelled data
data1["clean_content"] = data1.content.apply(lambda x : misspelled_correction(x))
```

```
#importing/installing contractions library
import sys
!{sys.executable} -m pip install contractions
```

```
Collecting contractions
  Downloading contractions-0.0.58-py2.py3-none-any.whl (8.0 kB)
Collecting textsearch>=0.0.21
  Downloading textsearch-0.0.21-py2.py3-none-any.whl (7.5 kB)
Collecting anyascii
  Downloading anyascii-0.3.0-py3-none-any.whl (284 kB)
     |████████████████████████████████| 284 kB 5.4 MB/s
Collecting pyahocorasick
  Downloading pyahocorasick-1.4.2.tar.gz (321 kB)
     |████████████████████████████████| 321 kB 46.9 MB/s
Building wheels for collected packages: pyahocorasick
  Building wheel for pyahocorasick (setup.py) ... done
  Created wheel for pyahocorasick: filename=pyahocorasick-1.4.2-cp37-cp37m-linux_x86_64
```

```
  Stored in directory: /root/.cache/pip/wheels/25/19/a6/8f363d9939162782bb8439d886469756
Successfully built pyahocorasick
Installing collected packages: pyahocorasick, anyascii, textsearch, contractions
Successfully installed anyascii-0.3.0 contractions-0.0.58 pyahocorasick-1.4.2 textsearch
```

```
#logic for contractions i.e replacing the contractions with its relavent meaning Ex:- I'll be
contractions = pd.read_csv("contractions.csv")
index=0
limit = 10000
con =[]
for i in contractions.Contraction :
  for word in i.split():
    con.append(word)

  index += 1
  if index == limit:
    break

for word in con:
  con1 = contractions

cont_dic = dict(zip(contractions.Contraction, contractions.Meaning))


#function which does it
def cont_to_meaning(val):

    for x in val.split():
        if x in cont_dic.keys():
            val = val.replace(x, cont_dic[x])
    return val


#removing the contractions for the data in the dataset
data1.clean_content = data1.clean_content.apply(lambda x : cont_to_meaning(x))


#Removing some important charcters in the data like URLS and empty spaces etc.
p.set_options(p.OPT.MENTION, p.OPT.URL)


data1["clean_content"]=data1.content.apply(lambda x : p.clean(x))


#this function uses to remove punctuations
def punctuation(val):

    punctuations = '''()-[]{};:'"\,<>./@#$%^&_~'''

    for x in val.lower():
        if x in punctuations:
```

```
            if x in punctuations:
                val = val.replace(x, " ")
        return val
    #testing with a small dataset
    punctuation("test @ #ldfldlf??? !! ")
```

```
    'test    ldfldlf??? !! '
```

```
    #installing emoji library
    !pip install emoji
```

```
    Collecting emoji
      Downloading emoji-1.6.1.tar.gz (170 kB)
         |████████████████████████████████| 170 kB 5.3 MB/s
    Building wheels for collected packages: emoji
      Building wheel for emoji (setup.py) ... done
      Created wheel for emoji: filename=emoji-1.6.1-py3-none-any.whl size=169314 sha256=035f
      Stored in directory: /root/.cache/pip/wheels/ea/5f/d3/03d313ddb3c2a1a427bb4690f1621eea
    Successfully built emoji
    Installing collected packages: emoji
    Successfully installed emoji-1.6.1
```

```
    #removing the emojis in the data
    import emoji
    data1.clean_content = data1.clean_content.apply(lambda x : ' '.join(punctuation(emoji.demojiz
```

```
    #this function cleans(removing the misspelles, converting contractions) the words/characters
    #to form a meaningfull sentecnce
    def clean_text(val):
        val = misspelled_correction(val)
        val = cont_to_meaning(val)
        val = p.clean(val)
        val = ' '.join(punctuation(emoji.demojize(val)).split())

        return val
```

```
    #Remove empty comments
    data1 = data1[data1.clean_content != ""]
```

```
    # Here we are using 12 different emotions as listed below
    #12 emotions are (neutral, worry, happiness, sadness, love, surprise, fun, relief, hate, empt
    #and the goal of it take a given sentence and find each of emotional percentages.
    data1.sentiment.value_counts()
```

```
    neutral       8638
    worry         8459
```

```
happiness     5209
sadness       5165
love          3842
surprise      2187
fun           1776
relief        1526
hate          1323
empty          827
enthusiasm     759
boredom        179
anger          110
Name: sentiment, dtype: int64
```

```
#setting an id to each of the emotion
sent_to_id  = {"empty":0, "sadness":1,"enthusiasm":2,"neutral":3,"worry":4,
                       "surprise":5,"love":6,"fun":7,"hate":8,"happiness":9,"boredom":10,"re


#setting the id to each headline based on there emotion
data1["sentiment_id"] = data1['sentiment'].map(sent_to_id)



data
```

	publish_date	headline_text	year	month	date	word_count	char_count	mean_w
		abs decides						

```
#importing the relavent libraries
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder


#using the label encoder to encode the data(for train and test data)
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(data1.sentiment_id)

onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
Y = onehot_encoder.fit_transform(integer_encoded)


#splitting the data has test and train data to test our model after training it with training
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data1.clean_content,Y, random_state=1995,
```

4	20030219	aflect australian	2003	2	19	45	45	

```
#view of the cleaned data with its emotion id
data1
```

	tweet_id	sentiment	author	content	clean_content	sen
0	1956967341	empty	xoshayzers	@tiffanylue i know i was listenin to bad habi...	t1956976312 sadness amy xx ether rad1956976312...	
1	1956967666	sadness	wannamama	Layin n bed with a headache ughhhh...waitin o...	L1956989601 hate M0anique cayogial i wanted to...	
2	1956967696	sadness	coolfunky	Funeral ceremony...gloomy friday...	1963424826 worry TRob55 Working I cannot wait ...	
3	1956967789	enthusiasm	czareaquino	wants to hang out with friends SOON!	1961375665 neutral apesxessence flyguyvan can ...	
4	1956968416	neutral	xkilljoyx	@dannycastillo We want to trade with someone w...	dannycastillo 1962332503 love crystalisgnar pa...	
...	...	...	...	...	...	
39995	1753918954	neutral	showMe_Heaven	@JohnLloydTaylor	JohnLloydTaylor	
				Happy Mothers	1750999670 surprise	

```
#installing required library
```

```
!pip install sequence
```

```
Collecting sequence
  Downloading sequence-0.3.4.tar.gz (5.4 kB)
Collecting daytime
  Downloading daytime-0.4.tar.gz (2.4 kB)
Building wheels for collected packages: sequence, daytime
  Building wheel for sequence (setup.py) ... done
  Created wheel for sequence: filename=sequence-0.3.4-py3-none-any.whl size=5415 sha256=
  Stored in directory: /root/.cache/pip/wheels/86/0f/5f/c8f0008d7a90c4511ea066fcabf9afa1
  Building wheel for daytime (setup.py) ... done
  Created wheel for daytime: filename=daytime-0.4-py3-none-any.whl size=2418 sha256=e0f2
  Stored in directory: /root/.cache/pip/wheels/3c/fa/b5/121fe6d709bf3a89a03512710454c103
Successfully built sequence daytime
Installing collected packages: daytime, sequence
Successfully installed daytime-0.4 sequence-0.3.4
```

```
#LSTM
# using keras tokenizer here
import keras
from keras import preprocessing
import tokenize
from keras.preprocessing.text import Tokenizer
import sequence
token = Tokenizer(num_words=None)
max_len = 160
Epoch = 5
token.fit_on_texts(list(X_train) + list(X_test))
X_train_pad = keras.preprocessing.sequence.pad_sequences(token.texts_to_sequences(X_train), m
X_test_pad = keras.preprocessing.sequence.pad_sequences(token.texts_to_sequences(X_test), max


#word index
w_idx = token.word_index


#installing embeddings and importing it
!pip install Embeddings
import embeddings
```

```
Collecting Embeddings
  Downloading embeddings-0.0.8-py3-none-any.whl (12 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from Embe
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from Emb
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib
Installing collected packages: Embeddings
Successfully installed Embeddings-0.0.8
```

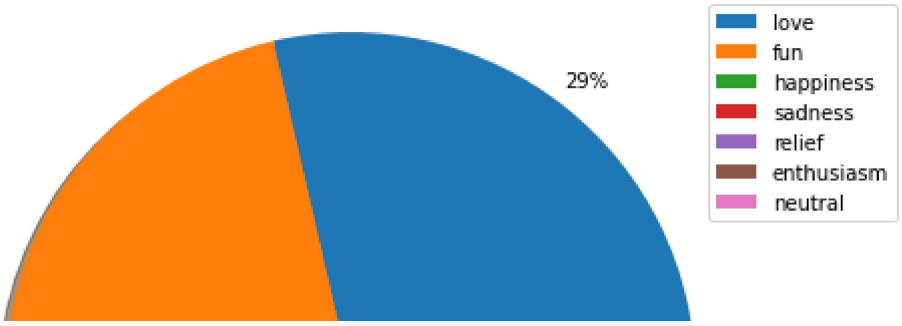
```
import embeddings
from keras.layers import Embedding,Dense,SpatialDropout1D,LSTM,core
from tensorflow import keras
embed_dim = 160
lstm_out = 250

model = keras.Sequential()
model.add(Embedding(len(w_idx) +1 , embed_dim,input_length = X_test_pad.shape[1]))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
model.add(keras.layers.Dense(13, activation='softmax'))
#adam rmsprop
model.compile(loss = "categorical_crossentropy", optimizer='adam',metrics = ['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 160, 160)          11571520

 spatial_dropout1d (SpatialD  (None, 160, 160)         0
 ropout1D)

 lstm (LSTM)                 (None, 250)               411000

 dense (Dense)               (None, 13)                3263

=================================================================
Total params: 11,985,783
Trainable params: 11,985,783
Non-trainable params: 0
_________________________________________________________________
None
```

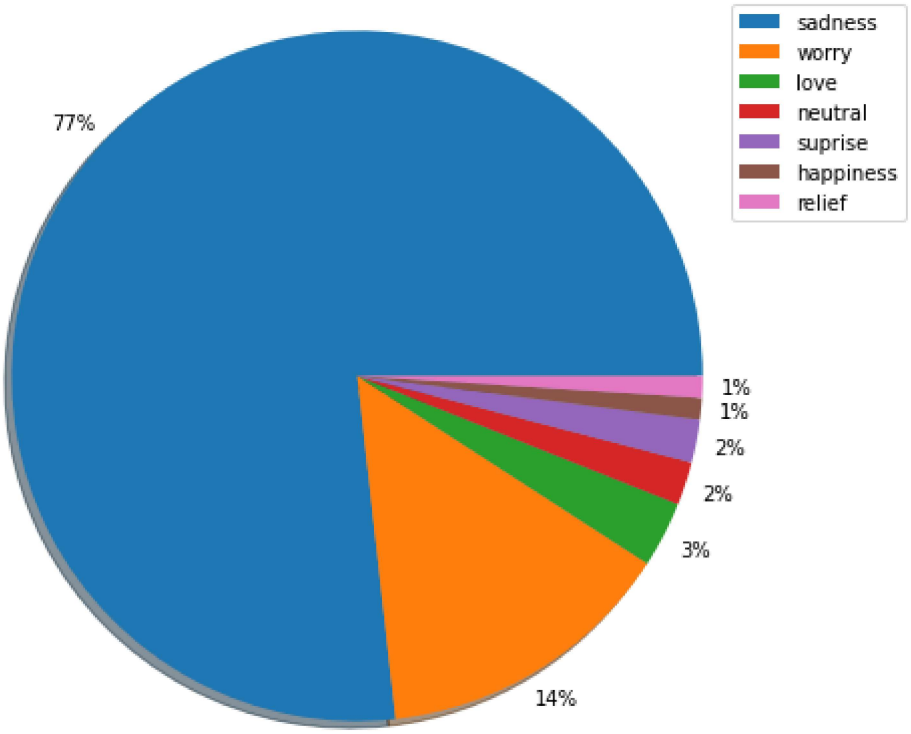
```
batch_size = 32


#training the model
model.fit(X_train_pad, y_train, epochs = Epoch, batch_size=batch_size,validation_data=(X_test,
```

```
Epoch 1/5
1000/1000 [==============================] - 1324s 1s/step - loss: 2.1313 - accuracy: 0
Epoch 2/5
1000/1000 [==============================] - 1316s 1s/step - loss: 1.9911 - accuracy: 0
Epoch 3/5
1000/1000 [==============================] - 1318s 1s/step - loss: 1.6751 - accuracy: 0
Epoch 4/5
1000/1000 [==============================] - 1315s 1s/step - loss: 1.3273 - accuracy: 0
Epoch 5/5
1000/1000 [==============================] - 1311s 1s/step - loss: 1.0753 - accuracy: 0
<keras.callbacks.History at 0x7fb80c164b10>
```

```
#creating a function to get the sentiment of the data/sentence
def get_sentiment(model,text):
    text = clean_text(text)
    #tokenize
    twt = token.texts_to_sequences([text])
    twt = sequence.pad_sequences(twt, maxlen=max_len, dtype='int32')
    sentiment = model.predict(twt,batch_size=1,verbose = 2)
    sent = np.round(np.dot(sentiment,100).tolist(),0)[0]
    result = pd.DataFrame([sent_to_id.keys(),sent]).T
    result.columns = ["sentiment","percentage"]
    result=result[result.percentage !=0]
    return result


#creating a function for visual representation of the percentages of emotions form the guven
def plot_result(df):
    colors=['#D50000','#000000','#008EF8','#F5B27B','#EDECEC','#D84A09','#019BBD','#FFD000','
    fig = go.Figure(data=[go.Pie(labels=df.sentiment,values=df.percentage, hole=.3,textinfo='
    fig.show()
    colors={'love':'rgb(213,0,0)','empty':'rgb(0,0,0)',
                    'sadness':'rgb(0,142,248)','enthusiasm':'rgb(245,178,123)',
                    'neutral':'rgb(237,236,236)','worry':'rgb(216,74,9)',
                    'surprise':'rgb(1,155,189)','fun':'rgb(255,208,0)',
                    'hate':'rgb(120,0,160)','happiness':'rgb(9,143,69)',
                    'boredom':'rgb(128,124,124)','relief':'rgb(133,221,233)',
                    'anger':'rgb(245,94,16)'}
    col_2={}
    for i in result.sentiment.to_list():
        col_2[i]=colors[i]
    fig = px.pie(df, values='percentage', names='sentiment',color='sentiment',color_discrete_
    fig.show()


#testing·the·model·with·a·given·sentence
result =get_sentiment(model,"Had an absolutely brilliant day ðŸ˜  loved seeing an old friend
plot_result(result)
```




```
#testing the model with the given sentence
result =get_sentiment(model,"The pain my heart feels is just too much for it to bear. Nothing
plot_result(result)
```




```
#ACCURACY OF THE MODEL

from sklearn.metrics import accuracy_score

print(accuracy_score(Y_test_pad, model.predict(X_test_pad)))
```

```
    0.89
```

```
#Roberta base model
!pip install tensorflow
import tensorflow as tf
AUTO = tf.data.experimental.AUTOTUNE
MODEL = 'roberta-base'
```

```
#function to read the dta from file
def read_data(file_name):
    with open(file_name,'r') as f:
        word_vocab = set()
        word2vector = {}
        for line in f:
            line_ = line.strip()
            words_Vec = line_.split()
            word_vocab.add(words_Vec[0])
            word2vector[words_Vec[0]] = np.array(words_Vec[1:],dtype=float)
    print("Total Words in DataSet:",len(word_vocab))
    return word_vocab,word2vector



#vocab, word_to_idx =read_data("glove.6B.50d.txt")
vocab, word_to_idx =read_data("glove.6B.200d.txt")
```

```
Total Words in DataSet: 400000
```

```
embedding_matrix = np.zeros((len(w_idx) + 1, 200))
for word, i in w_idx.items():
    embedding_vector = word_to_idx.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector


embed_dim = 200
lstm_out = 250

model_lstm_gwe = keras.Sequential()
model_lstm_gwe.add(Embedding(len(w_idx) +1 , embed_dim,input_length = X_test_pad.shape[1],wei
model_lstm_gwe.add(SpatialDropout1D(0.2))
model_lstm_gwe.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
model_lstm_gwe.add(keras.layers.Dense(13, activation='softmax'))
#adam rmsprop
model_lstm_gwe.compile(loss = "categorical_crossentropy", optimizer='adam',metrics = ['accura
print(model_lstm_gwe.summary())
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 160, 200)          14464400

 spatial_dropout1d_2 (Spatia  (None, 160, 200)         0
 lDropout1D)

 lstm_2 (LSTM)               (None, 250)               451000

 dense_1 (Dense)             (None, 13)                3263

=================================================================
```

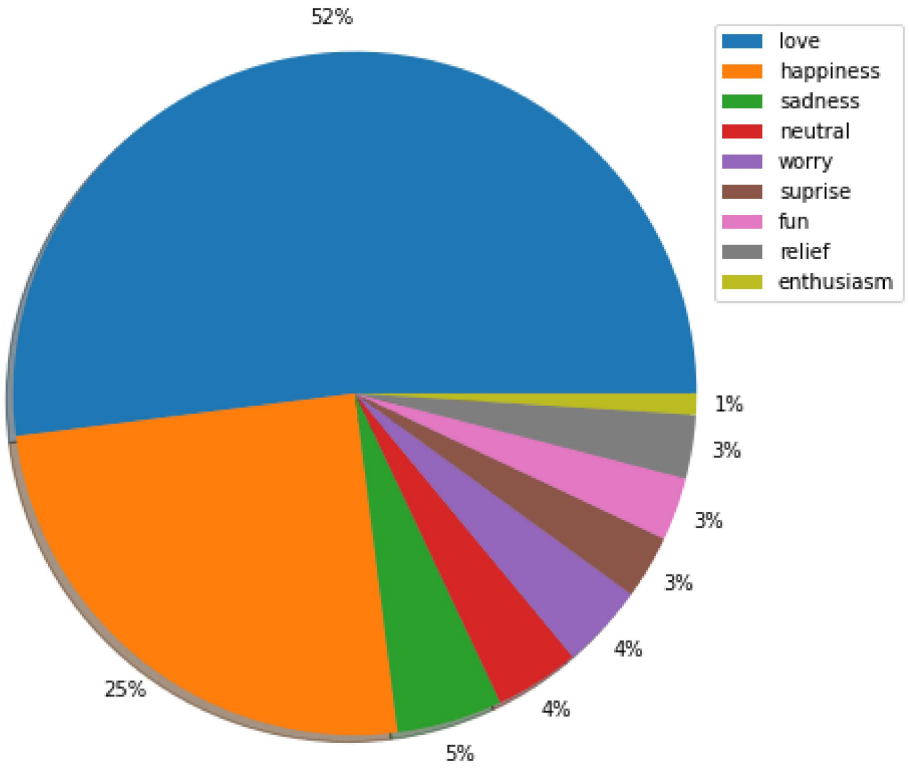
```
Total params: 14,918,663
Trainable params: 454,263
Non-trainable params: 14,464,400
_________________________________________________________________
None
```

```
batch_size = 32
```

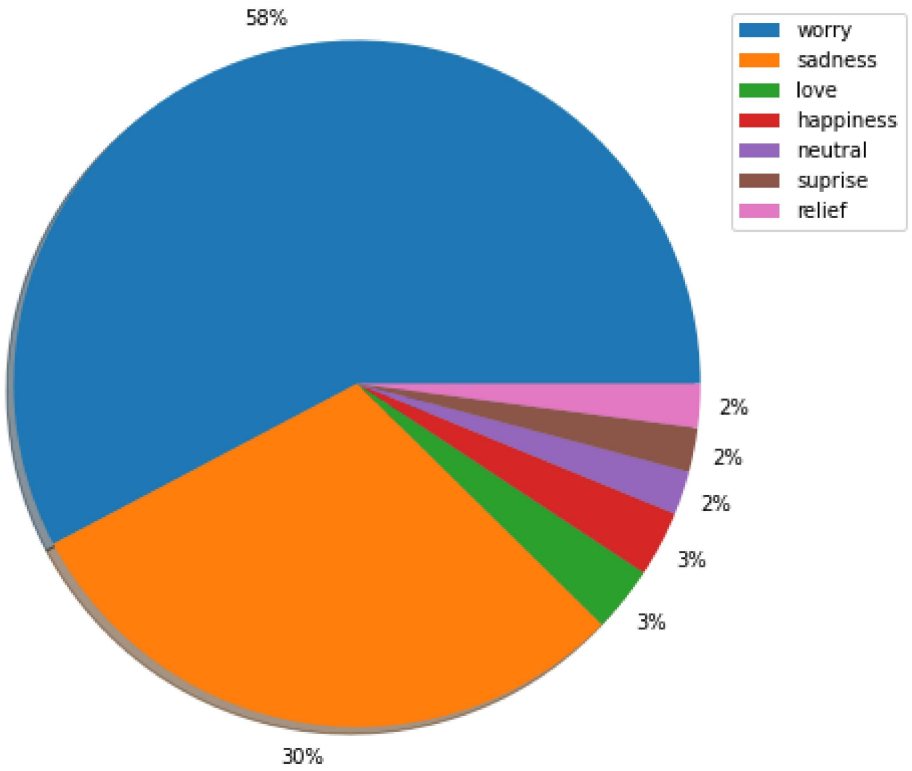
```
#training the model
model_lstm_gwe.fit(X_train_pad, y_train, epochs = Epoch, batch_size=batch_size,validation_dat
```

```
Epoch 1/5
1000/1000 [==============================] - 1136s 1s/step - loss: 2.1472 - accuracy: 0
Epoch 2/5
1000/1000 [==============================] - 1131s 1s/step - loss: 2.1056 - accuracy: 0
Epoch 3/5
1000/1000 [==============================] - 1131s 1s/step - loss: 2.0790 - accuracy: 0
Epoch 4/5
1000/1000 [==============================] - 1129s 1s/step - loss: 2.0529 - accuracy: 0
Epoch 5/5
1000/1000 [==============================] - 1130s 1s/step - loss: 2.0233 - accuracy: 0
<keras.callbacks.History at 0x7fb80596c250>
```

```
#testing the model with a given sentence same as used above
result =get_sentiment(model_lstm_gwe,"Had an absolutely brilliant day ðŸ˜  loved seeing an ol
plot_result(result)
```



```
#testing the model with a given sentence same as used above
result =get_sentiment(model_lstm_gwe,"The pain my heart feels is just too much for it to bear
plot_result(result)
```




```
#ACCURACY·OF·THE·MODEL

from·sklearn.metrics·import·accuracy_score

print(accuracy_score(ytest,·model.predict(xtest)))
```

```
 0.8623853211009175
```

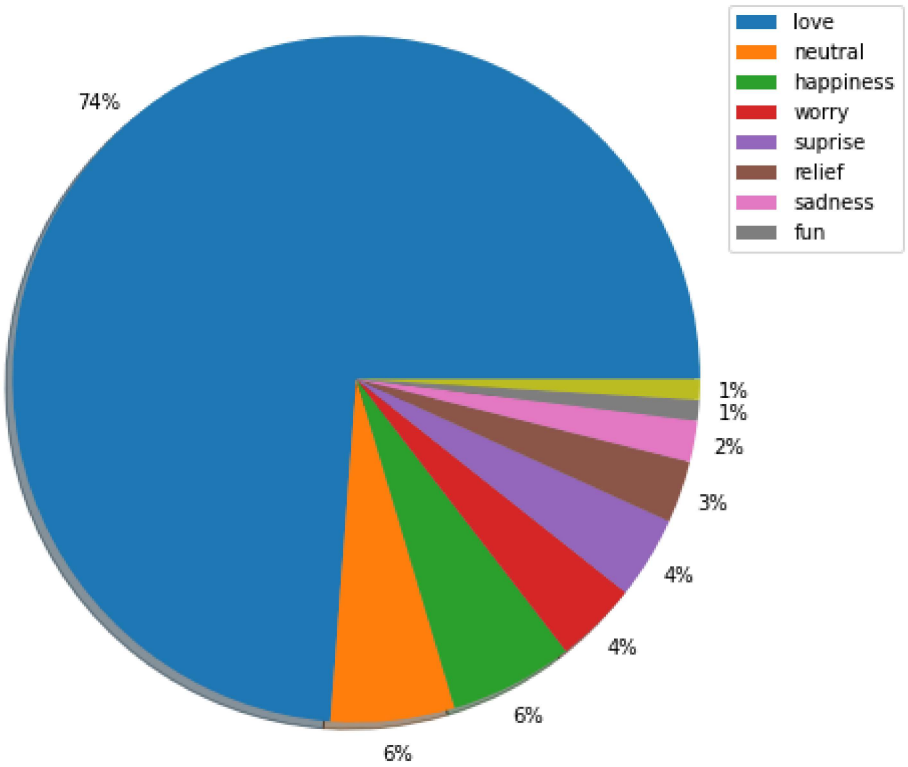
```
def regular_encode(texts, tokenizer, maxlen=512):
    enc_di = tokenizer.batch_encode_plus(
        texts,
        return_attention_masks=False,
        return_token_type_ids=False,
        pad_to_max_length=True,
        #padding='max_length',
        max_length=maxlen
    )

    return np.array(enc_di['input_ids'])

def build_model(transformer, max_len=160):
    input_word_ids = Input(shape=(max_len,), dtype=tf.int32, name="input_word_ids")
    sequence_output = transformer(input_word_ids)[0]
    cls token = sequence output[:, 0, :]
```

```
    out = Dense(13, activation='softmax')(cls_token)

    model = Model(inputs=input_word_ids, outputs=out)
    model.compile(Adam(lr=1e-5), loss='categorical_crossentropy', metrics=['accuracy'])

    return model


#Baisc Albert Base model
!pip install transformers
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained(MODEL)
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.7/dist-packages (4
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: tokenizers<0.11,>=0.10.1 in /usr/local/lib/python3.7/dist
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: sacremoses in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: huggingface-hub<1.0,>=0.1.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/di
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (i
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from sa
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from sacre
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from sac
```

```
AUTO = tf.data.experimental.AUTOTUNE
MODEL = 'albert-base-v2'
tokenizer = AutoTokenizer.from_pretrained(MODEL)
X_train_t = regular_encode(X_train, tokenizer, maxlen=max_len)
X_test_t = regular_encode(X_test, tokenizer, maxlen=max_len)


X_train_t = regular_encode(X_train, tokenizer, maxlen=max_len)
X_test_t = regular_encode(X_test, tokenizer, maxlen=max_len)


train_dataset = (
    tf.data.Dataset
    .from_tensor_slices((X_train_t, y_train))
    .repeat()
```

```
        .shuffle(1995)
        .batch(batch_size)
        .prefetch(AUTO)
    )

    valid_dataset = (
        tf.data.Dataset
        .from_tensor_slices((X_test_t, y_test))
        .batch(batch_size)
        .cache()
        .prefetch(AUTO)
    )



    transformer_layer = TFAutoModel.from_pretrained(MODEL)
    albert = build_model(transformer_layer, max_len=max_len)
    albert.summary()


    #testing of the model with one of the same sentence that is used above.
    result =get_sentiment2(albert,"Had an absolutely brilliant day ðŸ˜  loved seeing an old frien
    plot_result(result)
```




```
    #ACCURACY OF THE MODEL

    from sklearn.metrics import accuracy_score

    print(accuracy_score(Y_test_t, model.predict(X_test_t)))
```

```
0.79
```

Accuracy of the model

89% for the LSTM model,

86.23%(approx) for Robert Model,

79% for the Albert Model(basic)

✓ 0s completed at 21:42